AFRL-RI-RS-TR-2009-101
Final Technical Report
April 2009

INFORMATION INTEGRATION SEEDLING FOR DATA INTEGRATION AND EXPLOITATION SYSTEM THAT LEARNS (DIESEL)

SRI International

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2009-101 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/

ROGER J. DZIEGIEL, Jr.
Work Unit Manager

/s/

JOSEPH CAMERA, Chief
Information & Intelligence Exploitation Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* APR 09	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* Sep 07 – Dec 08

4. TITLE AND SUBTITLE

INFORMATION INTEGRATION SEEDLING FOR DATA INTEGRATION AND EXPLOITATION SYSTEM THAT LEARNS (DIESEL)

5a. CONTRACT NUMBER
FA8750-07-D-0185/0003

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
61101E

6. AUTHOR(S)

Pedro M. Domingos

5d. PROJECT NUMBER
PALT

5e. TASK NUMBER
QD

5f. WORK UNIT NUMBER
03

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025-3453

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RIED
525 Brooks Rd.
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2009-101

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 88ABW-2009-1333

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This Seedling investigated three key problems in information integration. All three problems are present simultaneously, and a truly robust and widely applicable information integration system therefore needs to solve the three problems simultaneously. A unified approach was developed to address these problems.

15. SUBJECT TERMS
Entity Resolution, Schema Matching, Concept Matching, Markov Logic, Algorithms, Artificial Intelligence, Machine Learning, Computer Science

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Roger J. Dziegiel, Jr.
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	25	**19b. TELEPHONE NUMBER (*Include area code)*** N/A

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

The goal of the University of Washington effort under DIESEL is to develop a unified approach to entity, schema and concept matching. *Entity resolution* is the problem of determining which mentions in the data correspond to the same object (e.g., “J. Smith” and “Jane Smith” may be the same person). *Schema matching* is the problem of determining which fields in a database or other structure correspond to the same attributes (e.g., “Contact” and “Telephone” may be the same attribute). *Concept matching* (a.k.a. ontology alignment) is the problem of determining which concepts in two taxonomies correspond to each other (e.g., “Faculty” in one taxonomy may mean the same as “Staff” in another). To date, each of these problems has been addressed separately, assuming that the other two have been solved *a priori* (e.g., schema matching may be performed assuming that objects and concepts have already been resolved). In most cases, however, all three problems are present simultaneously, and a truly robust and widely applicable information integration system therefore needs to solve the three simultaneously.

We successfully developed the approach we planned, as described in a series of papers [1, 2, 3], building on our earlier work on entity resolution [4, 5, 6]. Our approach uses Markov logic and a combination of existing and new learning and inference algorithms for it [7]. The key idea is to leverage *joint inference*, gradually propagating information from easier to harder matches. For example, if two fields are the same, then perhaps the corresponding objects are the same, and maybe the concepts they instantiate are also the same. We developed both supervised and unsupervised approaches (i.e., with and without labeled data), and observation-level and object-level approaches (i.e., inferring equality of observations vs. inferring their membership in objects, relations, etc.). Generally speaking, unsupervised object-level matching is the superior approach, and the one we would recommend *a priori*. We also studied the incorporation of background knowledge into the matching process, and we found that it is extremely helpful, in the sense that a small amount of easily-stated knowledge can go a long way toward ensuring accurate matching. We also studied how background knowledge can be efficiently induced from data if it is not known *a priori*, and found that the learned knowledge correctly captures the regularities in the data, and helps in ensuring good matches.

We begin by briefly reviewing some background on Markov logic. Then we describe the three systems we developed in detail, and present their experimental results. Finally we conclude with some recommendations.

2.0 BACKGROUND ON MARKOV LOGIC

Markov logic networks (MLNs) combine logic and probability by attaching weights to *first-order logic* rules [8], and viewing these as templates for features of *Markov networks* [9].

In first-order logic, formulas are constructed using four types of symbols: constants, variables, functions, and predicates. Constants represent objects in the domain of discourse (e.g., people: (*Anna*, *Bob*, etc.). Variables (e.g., *x*, *y*) range over the objects in the domain. Predicates represent relations among objects (e.g., *Friends*), or attributes of objects (e.g., *Student*). Variables and constants may be typed. An *atom* is a predicate symbol applied to a list of arguments, which may be variables or constants (e.g., *Friends(Anna,x)*). (In this report, we use *predicate* and *relation* interchangeably.) A *ground atom* is an atom all of whose arguments are constants (e.g., *Friends(Anna,Bob)*). A *world* is an assignment of truth values to all possible ground atoms. A database is a partial specification of a world; each atom in it is true, false or (implicitly) unknown. A clause is a disjunction of non-negated/negated atoms.

A Markov network or Markov random field is a model for the joint distribution of a set of variables $X = (X_1, \dots, X_n) \in \boldsymbol{X}$. It is composed of an undirected graph G and a set of potential functions ϕ_k. The graph has a node for each variable, and the model has a potential function for each clique in the graph. A potential function is a non-negative real-valued function of the state of the corresponding clique. The joint distribution represented by a Markov network is given by $P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}})$ where $\phi_k(x_{\{k\}})$ is the state of the kth clique (i.e., the state of the variables that appear in that clique). Z, known as the *partition function*, is given by $Z = \sum_{x \in \boldsymbol{X}} \prod_k \phi_k(x_{\{k\}})$. Markov networks are often conveniently represented as *log-linear models*, with each clique potential replaced by an exponentiated weighted sum of features of the state, leading to $P(X = x) = \frac{1}{Z} \exp(\sum_j w_j f_j(x))$. A feature may be any real-valued function of the state. This report will focus on binary features $f_j(x) \in \{0,1\}$. In the most direct translation from the potential-function form, there is one feature corresponding to each possible state $x_{\{k\}}$ of each clique, with its weight being $\log \phi_k(x_{\{k\}})$. This representation is exponential in the size of the cliques. However, we are free to specify a much smaller number of features (e.g., logical functions of the state of the clique), allowing for a more compact representation than the potential-function form, particularly when large cliques are present. Markov logic takes advantage of this.

A Markov logic network (MLN) is a set of weighted first-order formulas. Together with a set of constants representing objects in the domain, it defines a Markov network with one node per ground atom and one feature per ground formula. The weight of a feature is the weight of the first-order formula that originated it. The probability distribution over possible worlds x specified by the ground Markov network is given by $P(X = x) = \frac{1}{Z} \exp\left(\sum_{i \in F} \sum_{j \in G_i} w_i g_j(x)\right)$, where Z is the partition function, F is the set of all first-order formulas in the MLN, G_i is the set of groundings of the ith first-order formula, and $g_j(x) = 1$ if the jth ground formula is true and $g_j(x) = 0$ otherwise. Markov logic enables us to compactly represent complex models in non-i.i.d. domains. General algorithms for inference and learning in Markov logic are discussed in [7].

3.0 SEMANTIC NETWORK EXTRACTOR

3.1 System Description

Our Semantic Network Extractor (SNE) system [1] jointly clusters objects (entities) and relations (schemas/concepts) in an unsupervised manner, without requiring the number of clusters to be specified in advance. SNE does so by allowing information from object clusters it has created at each step to be used in forming relation clusters, and vice versa. The object clusters and relation clusters respectively form the nodes and links of a semantic network. A link exists between two nodes if and only if a true ground fact can be formed from the symbols in the corresponding relation and object clusters.

SNE is defined using finite second-order Markov logic in which variables can range over relations (predicates) as well as objects (constants). Extending Markov logic to second order involves simply grounding atoms with all possible predicate symbols as well as all constant symbols, and allows us to represent some models much more compactly than first-order Markov logic.

In SNE, we assume that relations are binary, i.e., relations are of the form $r(x,y)$ where r is a relation symbol, and x and y are object symbols. We use γ_i and Γ_i to respectively denote a cluster and clustering (i.e., a partitioning) of symbols of type i. If r, x, and y are respectively in cluster γ_r, γ_x, and γ_y, we say that $r(x,y)$ is in the *cluster combination* $(\gamma_r,\gamma_x,\gamma_y)$. The learning problem in SNE consists of finding the cluster assignment $\Gamma = \left(\Gamma_r,\Gamma_x,\Gamma_y\right)$ that maximizes the posterior probability $P(\Gamma|D) \propto P(\Gamma,D) = P(\Gamma)P(D|\Gamma)$ where D is a vector of truth assignments to the observable $r(x,y)$ ground atoms.

We define one MLN for the likelihood $P(D|\Gamma)$ component, and one MLN for the prior $P(\Gamma)$ component of the posterior probability with just four simple rules.

The MLN for the likelihood component only contains one rule stating that the truth value of an atom is determined by the cluster combination it belongs to:

$$\forall r,x,y,+\gamma_r,+\gamma_x,+\gamma_y \quad r \in \gamma_r \wedge x \in \gamma_x \wedge y \in \gamma_y \Rightarrow r(x,y)$$

The “+”' notation is syntactic sugar that signifies that there is an instance of this rule *with a separate weight* for each cluster combination $(\gamma_r,\gamma_x,\gamma_y)$. This rule predicts the probability of query atoms given the cluster memberships of the symbols in them. This is known as the *atom prediction* rule.

Three rules are defined in the MLN for the prior component. The first rule states that each symbol belongs to exactly one cluster:

$$\forall x\, \exists^1 \gamma \quad x \in \gamma$$

This rule is hard, i.e., it has infinite weight and cannot be violated.

The second rule imposes an exponential prior on the number of cluster combinations. This rule combats the proliferation of cluster combinations and consequent over fitting, and is represented by the formula

$$\forall \gamma_r,\gamma_x,\gamma_y \;\; \exists\, r,x,y \quad r \in \gamma_r \wedge x \in \gamma_x \wedge y \in \gamma_y$$

with negative weight $-\lambda$. The parameter λ is fixed during learning, and is the penalty in log-posterior incurred by adding a cluster combination to the model. Thus larger λs lead to fewer cluster combinations being formed. This rule represents the complexity of the model in terms of the number of instances of the atom prediction rule (which is equal to the number of cluster combinations).

The last rule encodes the belief that most symbols tend to be in different clusters. It is represented by the formula

$$\forall x, x', \gamma_x, \gamma_{x'} \;\; x \in \gamma_x \wedge x' \in \gamma_{x'} \wedge x \neq x' \Rightarrow \gamma_x \neq \gamma_{x'}$$

with positive weight μ. The parameter μ is also fixed during learning. We expect there to be many concepts and high-level relations in a large heterogeneous body of data. If the tuple extraction process samples instances of these concepts and relations sparsely, and we expect each concept or relation to have only a few instances sampled, in many cases only one. Thus we expect most pairs of symbols to be in different concept and relation clusters.

SNE simplifies the learning problem by performing hard assignment of symbols to clusters (i.e., instead of computing probabilities of cluster membership, a symbol is simply assigned to its most likely cluster). This allows the *maximum a posteriori (MAP)* weights of the atom prediction rules, and the MAP log-posterior to be computed in closed form. The equation for the log-posterior, as defined by the two MLNs, can be written in closed form as

$$\log P(\Gamma \mid R) = \sum_{k \in K} \left[t_k \log\left(\frac{t_k + \alpha}{t_k + f_k + \alpha + \beta}\right) + f_k \log\left(\frac{f_k + \beta}{t_k + f_k + \alpha + \beta}\right)\right] - \lambda m + \mu d + C \qquad (1)$$

where K is the set of cluster combinations; t_k and f_k are respectively the number of true and false ground atoms in cluster combination k; α and β are smoothing parameters; m is the number of cluster combinations, d is the number of pairs of symbols that belong to different clusters, and C is a constant.

Since the log-posterior can be computed in closed-form, SNE simply searches over cluster assignments, evaluating each assignment by its posterior probability. (To speed up the computation of Equation 2, we make an approximation to it. Please refer to [1] for details.) SNE uses a bottom-up agglomerative clustering algorithm to find the MAP clustering. The algorithm begins by assigning each symbol to its own unit cluster. Next we try to merge pairs of clusters of each type. We create candidate pairs of clusters, and for each of them, we evaluate the change in posterior probability (Eqn. 2) if the pair is merged. If the candidate pair improves posterior probability, we store it in a sorted list. We then iterate through the list, performing the best merges first, and ignoring those containing clusters that have already been merged. In this manner, we incrementally merge clusters until no merges can be performed to improve posterior probability. To avoid creating all possible candidate pairs of clusters of each type (which is quadratic in the number of clusters), we make use of canopies [10]. A canopy for relation symbols is a set of clusters such that there exist object clusters γ_x and γ_y, and for all clusters γ_r in the canopy, the cluster combination $(\gamma_r, \gamma_x, \gamma_y)$ contains at least one true ground atom $r(x, y)$. We say that the clusters in the canopy share the *property* $\left(\gamma_x, \gamma_y\right)$. Canopies for object symbols x and y are similarly defined. We only try to merge clusters in a canopy that is no larger than a parameter $CanopyMax$. This parameter limits the number of candidate cluster pairs we consider for merges, making our algorithm more tractable. Furthermore, by using canopies, we only try "good" merges, because symbols in clusters that share a property are more likely to belong to the same cluster than those in clusters with no property in common.

3.2 Experiments

We conducted experiments to investigate the efficacy of jointly clustering relations and objects vis-à-vis clustering them separately (i.e., clustering relations but not objects, and vice versa). We also investigated the effectiveness of SNE against three other relational clustering systems, viz., Multiple Relational Clusterings (MRC), Information-Theoretic Co-clustering (ITC), and Infinite Relational Model (IRM).

All experiments were conducted on a large Web dataset consisting of 2.1 million $r(x, y)$ triples (publicly available at http://knight.cis.temple.edu-/~yates/data/resolver_data.tar.gz) extracted in a Web crawl by the information extraction system TextRunner [11]. Each triple takes the form $r(x, y)$ where r is a relation symbol, and $x$and y are object symbols. Some example triples are: *named_after (Jupiter, Roman_god)* and *upheld (Court, ruling)*. There are 15,872 distinct symbols, 700,781 distinct x symbols, and 665,378 distinct $y$symbols. Two characteristics of TextRunner's extractions are that they are sparse and noisy. To reduce the noise in the dataset, we only considered symbols that appeared at least 25 times. This leaves 10,214 r symbols, 8942 x symbols, and 7995 y symbols. There are 2,065,045 triples that contain at least one symbol that appears at least 25 times. In all experiments, we set the $CanopyMax$ parameter to 50. We also made the closed-world assumption for all systems (i.e., all triples not in the dataset are assumed false). Because the other relational clustering systems do not scale to the Web dataset, we had to modify them to use SNE's search algorithm. We also limited MRC to find a single clustering (it is able to find multiple) for an apple-to-apple comparison with SNE.

We evaluated the clustering's learned by each model against a gold standard that we manually created. The gold standard assigns 2688 r symbols, 2568 x symbols, and 3058 y symbols to 874, 511, and 700 non-unit clusters respectively. We measured the pairwise precision, recall and F1 of each model against the gold standard. Pairwise precision is the fraction of symbol pairs in learned clusters that appear in the same gold clusters. Pairwise recall is the fraction of symbol pairs in gold clusters that appear in the same learned clusters. F1 is the harmonic mean of precision and recall.

Figure 1 shows a snippet of the semantic network learned by SNE. Table 1 shows the performance of SNE when it clusters relations and objects jointly and when it clusters them separately. From that figure, we can see that SNE has the better overall F1 when it clusters relations and objects jointly (SNE-Sep). We show the best F1s in bold. Table 2 compares performance of SNE to those of three other relational clustering systems, and shows that SNE has the best overall F1 score. From Table 3 which shows the runtimes of the various systems, we see that SNE scales well relative to the other systems. We also evaluated the systems in terms of the semantic statements that they learned where a semantic statement is a cluster combination with one true ground atom. We found that SNE outperforms the other systems in terms of the fraction of correct semantic statements discovered (see [1] for details). We also found the clusters discovered by SNE agree well with those in a publicly available ontology WordNet [12].

Table 1. Performance when SNE Clusters Relations and Objects Jointly and Separately (SNE-Sep)

	Relation			Object		
Systems	Precision	Recall	F1	Precision	Recall	F1
SNE	0.452	0.187	**0.265**	0.509	0.062	**0.110**
SNE-Sep	0.597	0.116	0.194	0.535	0.046	0.085

Table 2. Performance of SNE and Three Other Relational Clustering Systems

	Relation			Object		
Systems	Precision	Recall	F1	Precision	Recall	F1
SNE	0.452	0.187	**0.265**	0.509	0.062	**0.110**
IRM	0.201	0.089	0.124	0.280	0.042	0.073
ITC	0.773	0.003	0.006	0.617	0.025	0.048
MRC	0.054	0.044	0.049	0.045	0.009	0.015

Table 3. Runtimes of SNE and Three Other Relational Clustering Systems

Systems	Runtimes (hrs)
SNE	5.5
IRM	9.5
ITC	72.0
MRC	1.1

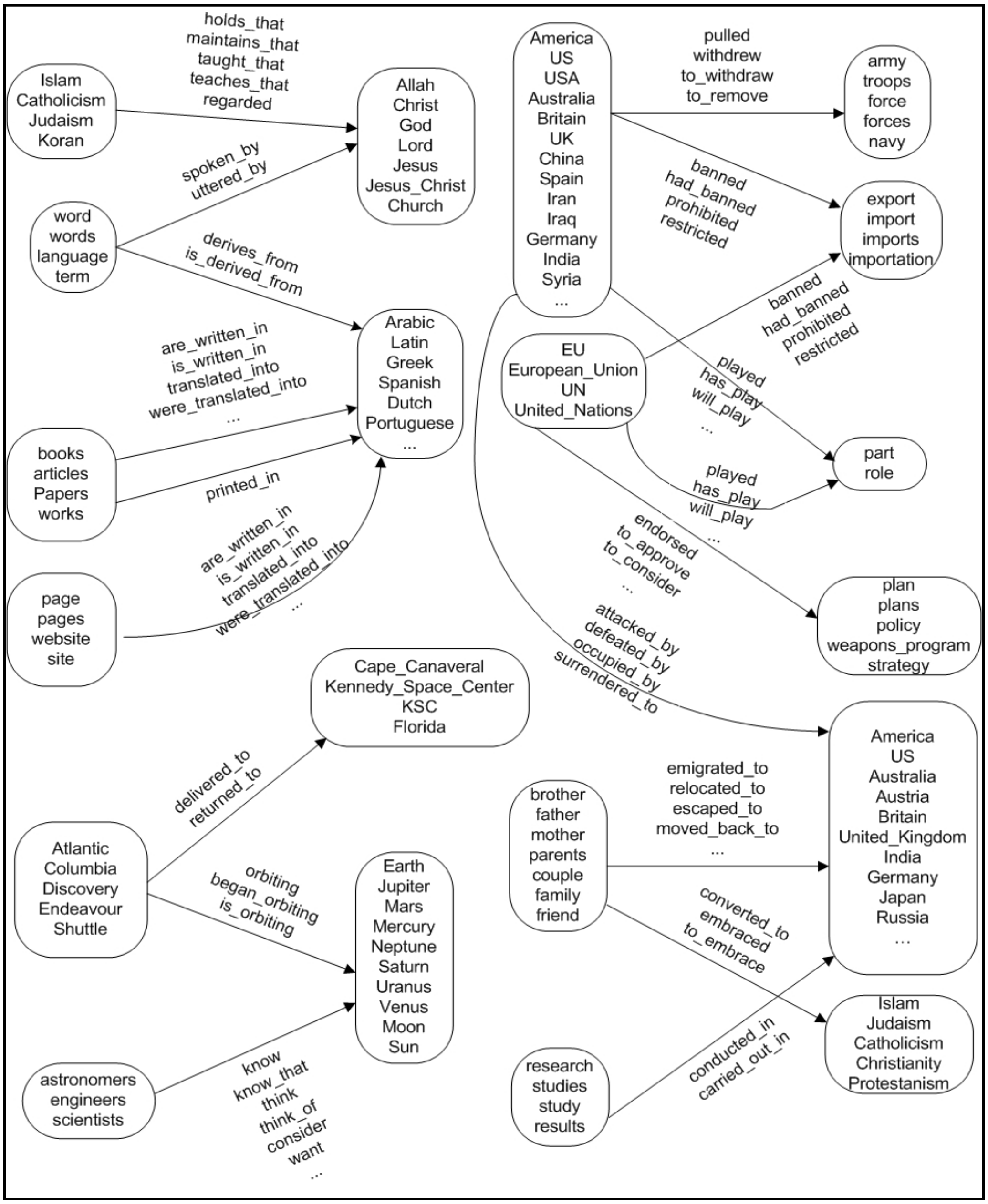
Islam Catholicism Judaism Koran
holds_that maintains_that taught_that teaches_that regarded
Allah Christ God Lord Jesus Jesus_Christ Church
word words language term
spoken_by uttered_by
derives_from is_derived_from
Arabic Latin Greek Spanish Dutch Portuguese ...
are_written_in is_written_in translated_into were_translated_into ...
books articles Papers works
printed_in
page pages website site
are_written_in is_written_in translated_into were_translated_into ...
America US USA Australia Britain UK China Spain Iran Iraq Germany India Syria ...
pulled withdrew to_withdraw to_remove
army troops force forces navy
banned had_banned prohibited restricted
export import imports importation
EU European_Union UN United_Nations
banned had_banned prohibited restricted
played has_play will_play ...
part role
played has_play will_play ...
endorsed to_approve to_consider ...
plan plans policy weapons_program strategy
attacked_by defeated_by occupied_by surrendered_to
Cape_Canaveral Kennedy_Space_Center KSC Florida
delivered_to returned_to
Atlantic Columbia Discovery Endeavour Shuttle
orbiting began_orbiting is_orbiting
Earth Jupiter Mars Mercury Neptune Saturn Uranus Venus Moon Sun
astronomers engineers scientists
know know_that think think_of consider want ...
America US Australia Austria Britain United_Kingdom India Germany Japan Russia ...
brother father mother parents couple family friend
emigrated_to relocated_to escaped_to moved_back_to ...
converted_to embraced to_embrace
Islam Judaism Catholicism Christianity Protestanism
research studies study results
conducted_in carried_out_in

Figure 1: Snippet of Semantic Network Learned by SNE

4.0 JOINT UNSUPERVISED COREFERENCE RESOLUTION

4.1 System Description

In this system, we demonstrate how we can easily add background knowledge using Markov logic to improve the matching of entities. We tested the efficacy of our system on the problem of *coreference resolution*, i.e., identifying *mentions* (typically noun phrases) that refer to the same *entities*. This is a key sub-problem in many natural language processing (NLP) applications, including information extraction, question answering, machine translation, etc.

Supervised learning approaches treat the problem as one of classification: for each pair of mentions, predict whether they corefer or not [13]. While successful, these approaches require labeled training data, consisting of mention pairs and the correct decisions for them. This limits their applicability. Unsupervised approaches are attractive due to the availability of large quantities of unlabeled text. However, unsupervised coreference resolution is much more difficult. The most sophisticated model to date proposed by [14] still lags supervised ones by a substantial margin. The lack of label information in unsupervised coreference resolution can potentially be overcome by performing joint inference, which leverages the "easy" decisions to help make related "hard" ones. Relations that have been exploited in supervised coreference resolution include transitivity and anaphor city. (Transitivity refers to the condition where if mentions A and B corefer, and B and C corefer, then A and C corefer. Anaphoricity refers to the condition where a linguistic unit (e.g., pronoun) refers back to another unit as in the use of *him* to refer to *Alan* in the sentence *Alan told Betty to get him some candy*.) However, there is little work to date on joint inference for unsupervised resolution. We address this problem using Markov logic, which allows us to easily build models involving relations among mentions, like apposition and predicate nominal's. By extending the state-of-the-art algorithms for inference and learning in Markov logic, we developed the first general-purpose unsupervised learning algorithm, and applied it to unsupervised coreference resolution.

We incrementally create more sophisticated MLNs for coreference resolution to illustrate the ease of specifying models in Markov logic.

4.1.1 Base MLN

The main query predicate is $InClust(m, c!)$, which is true if and only if mention m is in cluster c. (A query predicate is a predicate whose value we do not know at test time, would like to infer.) The "$c!$" notation signifies that for each m, this predicate is true for a unique value of c. The main evidence predicate is $Head(m, t!)$, where m is a mention and t a token, and which is true if and only if t is the head of m. A key component in our MLN is a simple head mixture model, where the mixture component priors are represented by the unit clause $InClust(+m, +c)$ and the head distribution is represented by the head prediction rule

$$InClust(m, +c) \ \wedge Head(m, +t).$$

All free variables are implicitly universally quantified. The "$+$" notation signifies that the MLN contains an instance of the rule, with a separate weight, for each value combination of the variables with a plus sign. By convention, at each inference step we name each non-empty cluster after the earliest mention it contains. This helps break the symmetry among mentions, which otherwise produces multiple optima and makes learning unnecessarily harder. To encourage clustering, we impose an exponential prior on the number of non-empty clusters with weight −1. The above model only clusters mentions with the same head, and does not work well for pronouns. To address this, we introduce the predicate $IsPrn(m)$, which is true if and only if the mention m is a pronoun, and adapt the head prediction rule as follows:

$$\neg IsPrn(m) \wedge InClust(m, +c) \wedge Head(m, +t)$$

This is always false when m is a pronoun, and thus applies only to non-pronouns. Pronouns tend to resolve with mentions that are semantically compatible with them. Thus we introduce predicates that represent entity type, number, and gender: $Type(x, e!)$, $Number(x, n!)$, $Gender(x, g!)$, where x can be either a cluster or mention, $n \in \{Singular, Plural\}$, $e \in \{Person, Organization, Location, Other\}$, and $g \in \{Male, Female, Neuter\}$. Many of these are known for pronouns, and some can be inferred from simple linguistic cues (e.g., "*Ms. Galen*" is a singular female person, while "*XYZ Corp.*" is an organization). (We used the following cues: *Mr.*, *Ms.*, *Jr.*, *Inc.*, *Corp.*, *corporation*, and *company*.) Entity type assignment is represented by the unit clause $Type(+x, +e)$, and similarly for number and gender. A mention should agree with its cluster in entity type. This is ensured by the hard rule (which has infinite weight and must be satisfied)

$$InClust(m, c) \Rightarrow (Type(m, e) \Leftrightarrow Type(c, e))$$

There are similar hard rules for number and gender.

Different pronouns prefer different entity types, as represented by

$$IsPrn(m) \wedge InClust(m, c) \wedge Head(m, +t) \wedge Type(c, +e)$$

which only applies to pronouns, and whose weight is positive if pronoun t is likely to assume entity type e and negative otherwise. There are similar rules for number and gender. Aside from semantic compatibility, pronouns tend to resolve with nearby mentions. To model this, we impose an exponential prior on the distance (number of mentions) between a pronoun and its antecedent, with weight −1.

4.1.2 Full MLN

Syntactic relations among mentions often suggest coreference. Incorporating such relations into our MLN is straightforward. We illustrate this with two examples: apposition and predicate nominals. We introduce a predicate for apposition, $Appo(x,y)$, where x, y are mentions, and which is true if and only if y is an appositive of x. We then add the rule

$$Appo(x,y) \Rightarrow (InClust(x,c) \Leftrightarrow \ InClust(y,c))$$

which ensures that x, y are in the same cluster if y is an appositive of x. Similarly, we introduce a predicate for predicate nominals, $PredNom(x,y)$, and the corresponding rule. The weights of both rules can be learned from data with a positive prior mean. For simplicity, in this paper we treat them as hard constraints.

4.1.3 Extensions to Weight Learning and Inference

In order to apply existing Markov logic inference and learning algorithms to the problem of unsupervised coreference resolution, we had to extend them. Unsupervised learning in Markov logic maximizes the conditional log-likelihood

$$L(x,y) = \log P(Y = y|X = x) = \log \sum_{z} P(Y = y, Z = z|X = x)$$

where Z are unknown predicates. In our coreference resolution MLN, Y includes $Head$ and known groundings of $Type$, $Number$ and $Gender$; Z includes $InClust$ and unknown groundings of $Type$, $Number$, $Gender$; and X includes $IsPrn$, $Appo$ and $PredNom$. (For simplicity, from now on we drop x from the formula.) With Z, the optimization problem is no longer convex. However, we can still find a local optimum using gradient descent, with the gradient being

$$\frac{\partial}{\partial w_i} L(y) = \mathrm{E}_{Z|y}[n_i] - \ \mathrm{E}_{Y,Z}[n_i]$$

where n_i is the number of true groundings of the i^{th} clause. We extended PSCG for unsupervised learning. The gradient is the difference of two expectations, each of which can be approximated using samples generated by MC-SAT [15]. The (i,j)th entry of the Hessian is now

$$\frac{\partial^2}{\partial w_i \partial w_j} L(y) = \mathrm{Cov}_{Z|y}\left[n_i, n_j\right] - \ \mathrm{Cov}_{Y,Z}[n_i, n_j]$$

and the step size can be computed accordingly. Since our problem is no longer convex, the negative diagonal Hessian may contain zero or negative entries, so we first took the absolute values of the diagonal and added 1, then used the inverse as the preconditioner. Notice that when the objects form independent subsets (in our cases, mentions in each document), we can process them in parallel and then gather sufficient statistics for learning. We developed an efficient parallelized implementation of our unsupervised learning algorithm using the message-passing interface (MPI). To reduce burn-in time, we initialized MC-SAT with the state returned by MaxWalkSAT [16], rather than a random solution to the hard clauses. In the existing implementation in Alchemy [17], SampleSAT [18] flips only one atom in each step, which is inefficient for predicates with unique-value constraints (e.g., $Head(m, c!)$). Such predicates can be viewed as multi-valued predicates (e.g., $Head(m)$ with value ranging over all c's) and are prevalent in NLP applications. We adapted SampleSAT to flip two or more atoms in each step so that the unique-value constraints are automatically satisfied. By default, MC-SAT treats each ground clause as a separate factor while determining the slice. This can be very inefficient for highly correlated clauses. For example, given a non-pronoun mention m currently in cluster c and with head t, among the mixture prior rules involving m $InClust(m, c)$ is the only one that is satisfied, and among those head-prediction rules involving m, $\neg IsPrn(m) \wedge InClust(m, c) \wedge Head(m, t)$ is the only one that is satisfied; the factors for these rules multiply to $\phi = \exp(w_{m,c} + w_{m,c,t})$, where $w_{m,c}$ is the weight for $InClust(m, c)$, and $w_{m,c,t}$ is the weight for $\neg IsPrn(m) \wedge InClust(m, c) \wedge Head(m, t)$, since an unsatisfied rule contributes a factor of $e^0 = 1$. We extended MC-SAT to treat each set of mutually exclusive and exhaustive rules as a single factor. E.g., for the above m, MC-SAT now samples u uniformly from $(0, \phi)$, and requires that in the next state ϕ be no less than u. Equivalently, the new cluster and head for m should satisfy $w_{m,c'} + w_{m,c',t'} \geq \log(u)$. We extended SampleSAT so that when it considers flipping any variable involved in such constraints (e.g., c or t above), it ensures that their new values still satisfy these constraints. The final clustering is found using the MaxWalkSAT weighted satisfiability solver, with the appropriate extensions. We first ran a MaxWalkSAT pass with only finite-weight formulas, then ran another pass with all formulas. We found that this significantly improved the quality of the results that MaxWalkSAT returned.

4.2 Experiments

We tested our approach on MUC-6, ACE-2004 and ACE Phrase-2 (ACE-2).The MUC-6 dataset consists of 30 documents for testing and 221 for training. The English version of the ACE-2004 training corpus contains two sections, BNEWS and NWIRE, with 220 and 128 documents, respectively. ACE-2 contains a training set and a test set. In our experiments, we only used the test set, which contains three sections, BNEWS, NWIRE, and NPAPER, with 51, 29, and 17 documents, respectively. We emphasize that our approach is unsupervised, and thus the data only contains raw text plus true mention boundaries. We evaluated our systems using two commonly-used scoring programs: MUC [19] and B^3 [20]. On MUC-6, we compared against the published results of the state-of-the-art unsupervised system by [14] (H&K), and against the state-of-the-art *supervised* system by [13] (M&W). On ACE-2004, we compared against the published results of H&K. On ACE-2, we compared against the published results of two *supervised* systems [21] (Ng) and [22] (D&B).

Table 4 shows the results on the MUC-6 and ACE-2004 datasets. Our approach (MLN) outperforms both H&K and M&W in precision, recall and F1. Table 5 and 6 shows the results on the ACE-2 dataset. Our approach outperforms Ng and is competitive with D&B on all measures.

Table 4. Coreference Results in MUC Scores on the MUC-6 and ACE-2004 datasets

Systems	MUC-6			ACE-2004 EN-BNEWS			ACE-2004 EN-NWIRE		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
H&K	80.4	62.4	70.3	63.2	61.3	62.3	66.7	62.3	64.2
M&W	-	-	73.4	-	-	-	-	-	-
MLN	83.0	75.8	79.2	66.8	67.8	67.3	71.3	70.5	70.9

Table 5. Coreference Results in MUC Scores on the ACE-2 datasets

Systems	BNEWS			NWIRE			NPAPER		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Ng	67.9	62.2	64.9	60.3	50.1	54.7	71.4	67.4	69.3
D&B	78.0	62.1	69.2	75.8	60.8	67.5	77.6	68.0	72.5
MLN	68.3	66.6	67.4	67.7	67.3	67.4	69.2	71.7	70.4

Table 6. Coreference Results in B^3 Scores on the ACE-2 datasets

Systems	BNEWS			NWIRE			NPAPER		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Ng	77.1	57.0	65.6	75.4	59.3	66.4	75.4	59.3	66.4
MLN	70.3	65.3	67.7	74.7	68.8	71.6	70.0	66.5	68.2

5.0 LEARNING MLN STRUCTURE VIA HYPERGRAPH LIFTING

5.1 System Description

We create the Learning via Hyperpgraph Lifting (LHL) system [3] to learn background knowledge (in the form of Markov logic rules) from data when it is not known *a priori*. Such knowledge could then be used for matching entities, schemas and concepts (as in the previous system).

Learning Markov logic rules and their associated weights is the problem of *MLN Structure Learning*. To date, most MLN structure learners [23, 24] systematically enumerate candidate clauses by starting from an empty clause, greedily adding literals to it, and testing the resulting clause's empirical fit to training data. Such a strategy has two shortcomings: searching the large space of clauses is computationally expensive; and it is susceptible to converging to a local optimum, missing potentially useful clauses. These shortcomings can be ameliorated by using the data to *a priori* constrain the space of candidates. This is the basic idea in *relational pathfinding* [25], which finds paths of true ground atoms that are linked via their arguments and then generalizes them into first-order rules. Each path corresponds to a conjunction that is true at least once in the data. Since most conjunctions are false, this helps to concentrate the search on regions with promising rules. However, pathfinding potentially amounts to exhaustive search over an exponential number of paths. Hence, systems using relational pathfinding typically restrict themselves to very short paths, creating short clauses from them and greedily joining them into longer ones.

Our system LHL uses relational pathfinding to a fuller extent than previous ones. It mitigates the exponential search problem by first inducing a more compact representation of data, in the form of a hypergraph over clusters of constants. Pathfinding on this 'lifted' hypergraph is typically at least an order of magnitude faster than on the ground training data, and produces MLNs that are more accurate.

A hypergraph is a straightforward generalization of a graph in which an edge can link any number of nodes, rather than just two. More formally, we define a hypergraph as a pair (V, E) where V is a set of nodes, and E is a multiset of labeled non-empty ordered subsets of V called hyperedges. In LHL, we find paths in a hypergraph. A path is defined as a set of hyperedges such that for any two hyperedges e_0 and e_n in the set, there exists an ordering of (a subset of) hyperedges in the set $e_0, e_1, \ldots, e_{n-1}, e_n$ such that e_n and e_{n+1} share at least one node.

A database can be viewed as a hypergraph with constants as nodes, and true ground atoms as hyperedges. Each hyperedge is labeled with a predicate symbol. Nodes (constants) are linked by a hyperedge (true ground atom) if and only if they appear as arguments in the hyperedge. (Henceforth we use *node* and *constant* interchangeably, and likewise for *hyperedge* and *true ground atom*.) A path of hyperedges can be generalized into a first-order clause by variabilizing their arguments. To avoid tracing the exponential number of paths in the hypergraph, LHL first jointly clusters the nodes into higher-level concepts, and by doing so it also clusters the hyperedges (i.e., the ground atoms containing the clustered nodes). The 'lifted' hypergraph has fewer nodes and hyperedges, and therefore fewer paths, reducing the cost of finding them.

Figure 2 provides an example. We have a database describing an academic department where professors tend to have students whom they are advising as teaching assistants (TAs) in the classes the professors are teaching. The left graph is created from the database, and after lifting, results in the right graph. Observe that the lifted graph is simpler and the clustered constants correspond to the high-level concepts of *Professor*, *Student*, and *Course*.

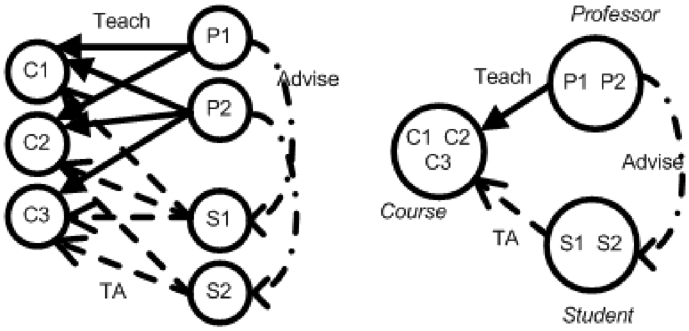

Figure 2: Example of Hypergraph Lifting

LHL consists of three steps. LHL begins by lifting a hypergraph. Then it finds paths in the lifted hypergraph. Finally it creates candidate clauses from the paths, and learn their weights to create an MLN. We describe each step in turn.

5.1.1 Hypergraph Lifting

We call our hypergraph lifting algorithm LiftGraph. LiftGraph is defined using similar Markov logic rules as SNE. It differs from SNE in the following ways. LiftGraph can handle relations of arbitrary arity, whereas SNE can only handle binary relations. While SNE can cluster relation symbols, in this report, for simplicity, LiftGraph do not cluster relations. (However, it is straightforward to extend LiftGraph to do so.) LiftGraph works by jointly clustering the constants in a hypergraph in a bottom-up agglomerative manner, allowing information to propagate from one cluster to another as they are formed. The number of clusters need not be pre-specified. As a consequence of clustering the constants, the ground atoms in which the constants appear are also clustered. Each hyperedge in the lifted hypergraph contains at least one true ground atom.

We use the same notation as SNE. In addition, we use $r(\gamma_1, \ldots, \gamma_n)$ to denote a hyperedge connecting nodes $\gamma_1, \ldots, \gamma_n$.A hypergraph representing the true ground atoms $r(x, \ldots, x_n)$ in a database is simply $(V = \{\{x_i\}\}, E = \{r(\{x_1\}, \ldots, \{x_n\}\})$ with each constant x_i in its own cluster, and a hyperedge for each true ground atom.

LiftGraph simplifies the learning problem by performing hard assignment of constant symbols to clusters (like SNE). The log-posterior of the LiftGraph model can now be computed in closed form. LiftGraph thus simply searches over cluster assignments, evaluating each one by its posterior probability. It begins by assigning each constant symbol x_i to its own cluster $\{x_i\}$, and creating a hyperedge $r(\{x_1\}, \ldots, \{x_n\})$ for each true ground atom $r(x_1, \ldots, x_n)$. Next it creates candidate pairs of clusters of each type, and for each pair, it evaluates the gain in posterior probability if its clusters are merged. It then chooses the pair that gives the largest gain to be merged. When clusters γ_i and γ_i' are merged to form new γ_i^{new}, each hyperedge $r(\gamma_1, \dots, \gamma_i, \dots \gamma_n)$ is replaced with $r(\gamma_1, \dots, \gamma_i^{new}, \dots \gamma_n)$ (and similarly for hyperedges containing γ_i'). Since $r(\gamma_1, \dots, \gamma_i, \dots \gamma_n)$ contains at least one true ground atom, $r(\gamma_1, \dots, \gamma_i^{new}, \dots \gamma_n)$ must do too. In this manner, LiftGraph incrementally merges clusters until no merges can be performed to improve posterior probability. It then returns a lifted hypergraph whose hyperedges all contain at least one true ground atom.

5.1.2 Path Finding

LHL constructs paths by starting from each hyperedge in a hypergraph. It begins by adding a hyperedge to an empty path, and then recursively adds hyperedges linked to nodes already present in the path (hyperedges already in the path are not re-added). Its search terminates when the path reaches a maximum length or when no new hyperedge can be added. Each time a hyperedge is added to the path, FindPath stores the resulting path as a new one. All the paths are passed on to the next step to create clauses.

5.1.3 Clause Creation and Pruning

A path in the hypergraph corresponds to a conjunction of $r(\gamma_1, \dots, \gamma_n)$ hyperedges, and it guarantees that the conjunction has at least one support in the hypergraph. We replace each γ_i in a path with a variable, thereby creating a variabilized atom for each hyperedge. We convert the conjunction of positive literals to a clause because that is the form that is typically used by ILP and MLN structure learning and inference algorithms usually. In Markov logic, a conjunction of positive literals with weight w is equivalent to a clause of negative literals with weight $-w$. In addition, we add clauses with the signs of up to n literals flipped (where n is a user-defined parameter), since the resulting clauses may also be useful. We evaluate each clause using weighted pseudo-log-likelihood (WPLL) [23].

We iterate over the clauses from shortest to longest. For each clause, we compare its scores against those of its sub-clauses (considered separately) that have already been retained. If the clause scores higher than all of these sub-clauses, it is retained; otherwise, it is discarded. In this manner, we discard clauses which are unlikely to be useful. Note that this process is efficient because the score of a clause only needs to be computed once, and can be cached for future comparisons. (Alternatively, we could evaluate a clause against all its sub-clauses taken together, but this would require re-optimizing the weights for each combination of sub-clauses for every comparison, which is computationally expensive.)

Finally we add the retained clauses to an MLN. We have the option of doing this in several ways. We could greedily add the rules one at a time in order of decreasing score. After adding each rule, we relearn the weights, and keep the rule in the MLN if it improves the overall WPLL. Alternatively, we could add all the rules to the MLN, and learn weights using L1 regularization to prune away ‘bad’ rules by giving them zero weights [26]. Lastly, we could use L2-regularization instead if the number of rules is not too large, and rely on the regularization to give ‘bad’ rules low weight. Optionally, we discard rules containing ‘dangling’ variables (i.e., variables which only appear once in a clause), since these are unlikely to be useful.

5.2 Experiments

We carried out experiments to investigate the performance of LHL on three datasets, publicly available at http://alchemy.cs.washington.edu. The IMDB dataset was created by [24] from the IMDB.com database. It describes a movie domain, and contains predicates describing movies, actors, directors, and their relationships (e.g, *WorkedIn(person,movie)*, etc.) The UW-CSE dataset, prepared by [7], describes an academic department. Its predicates describe students, faculty, and their relationships (e.g, *AdvisedBy(person1,person2)*, etc.). The Cora dataset, originally created by Andrew McCallum, is a collection of citations to computer science papers. Predicates include: *SameCitation(c1,c2)*, *TitleHasWord(title,word)*, etc. The IMDB, UW-CSE, and Cora datasets respectively have 17,793, 260,254, and 687,422 ground atoms, of which 1224, 2112, and 42,558 are true. Each dataset is divided into 5 folds. Note that the primary task in the Cora domain is the matching of entities, i.e., the citations, and their author, title and venue fields.

We compared LHL to two state-of-the-art systems: BUSL [24] and MSL [23]. Both systems are implemented in the Alchemy software package [17]. BUSL uses a form of relational pathfinding to find a path of ground atoms in the training data, but restricts itself to very short paths (length 2) to avoid fully searching the large space of paths. It then greedily pieces the path together into longer ones. MSL uses beam search to search for clauses. It begins from an empty clause, and systematically generates literals that can be used to extend the clause, evaluating each clause thus created for its empirical adequacy. The best clause it finds is added to an MLN, and the process is repeated until no new clauses can be found that improves the MLN's fit to data.

We evaluated the performance of the systems according to how well they predict the groundings of each predicate given groundings of all other predicates as evidence. For each dataset, we performed cross-validation using the five previously defined folds. To evaluate the performance of the systems, we measured the average conditional log-likelihood of the test atoms (CLL), and the area under the precision-recall curve (AUC). Table 7 shows the results of the systems. From the table, we see that LHL beats BUSL and MSL on 3 AUC and 2 CLL scores, but does worse on 1 CLL score. The runtimes of the systems also suggest that LHL scales better than the other systems.

Table 7. Experimental Comparison of MLN Structure Learners

Systems	IMDB			UW-CSE			Cora		
	AUC	CLL	Time(min)	AUC	CLL	Time (hr)	AUC	CLL	Time (hr)
LHL	0.73	-0.13	15.3	0.22	0.04	7.3	0.72	-0.64	13.6
BUSL	0.47	-0.14	4.7	0.21	0.05	12.9	0.17	-0.37	18.7
MSL	0.41	-0.18	0.2	0.18	0.57	2.1	0.17	-0.37	65.6

6.0 CONCLUSION AND RECOMMENDATION

We successfully developed the approach we planned. Our SNE system for extracting semantic networks from text is, to our knowledge, the most advanced to date in the scale and accuracy of the entity, schema and concept matching it can perform. Our unsupervised, object-level approach is currently the state of the art for coreference matching on standard datasets, outperforming even previous supervised approaches. Generally speaking, unsupervised object-level matching is the superior approach, and the one we would recommend *a priori*. Our LHL system for learning background knowledge from data when the knowledge is not available *a priori* also outperforms two state-of-the-art systems.

We originally planned to experiment on a variety of unstructured, semi-structured and structured data (e.g., free text, Web pages and databases, respectively). However, our experiments focused mainly on unstructured data, due to the difficulty in obtaining good semi-structured and structured datasets. Although the latter are of course common in the real world, there are currently no standard testbeds available that simultaneously include entity, schema and concept matching problems. This is not surprising, since research had previously not progressed this far, but a supposedly-available dataset we were planning to use turned out not to be.

Our work is important to data integration in general and DARPA in particular because it is the first to jointly handle the problems of entity, schema and concept matching. Since all three are usually present in real-world domains, truly effective data integration cannot be accomplished without it.

7.0 REFERENCES

[1] Kok, S. & Domingos, P., “Extracting Semantic Networks from Text via Relational Clustering”, *Proceedings of the Nineteenth European Conference on Machine Learning*, Antwerp, Belgium, 2008.

[2] Poon, H. & Domingos, P., “Joint Unsupervised Coreference Resolution with Markov Logic”, *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, Honolulu, HI, 2008.

[3] Kok, S. & Domingos, P., “Learning Markov Logic Network Structure via Hypergraph Lifting”, 2008, (in submission).

[4] Singla, P. & Domingos, P., “Entity Resolution with Markov Logic”, *Proceedings of the Sixth IEEE International Conference on Data Mining*, Hong Kong, 2006.

[5] Poon, H. & Domingos, P., “Joint Inference in Information Extraction”, *Proceedings of the Twenty Second National Conference on Artificial Intelligence*, Vancouver, Canada, 2007.

[6] Kok, S. & Domingos, P., “Statistical Predicate Invention”, *Proceedings of the Twenty Fourth International Conference on Machine Learning*, Corvallis, Oregon, 2007.

[7] Richardson, M. & Domingos, P., “Markov Logic Networks”, *Machine Learning*, **62**, 2006 pp. 107-136.

[8] Genesereth, M. R. & Nilsson, N. J., **Logical Foundations of Artificial Intelligence,** Morgan Kaufmann, San Mateo, CA, 1987.

[9] Pearl, J., **Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference**, Morgan Kaufmann, San Francisco, CA, 1988.

[10] McCallum, A.; Nigam, K. and Ungar, L., “Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching”, *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

[11] Banko, M.; Cafarella, M. J.; Soderland, S.; Broadhead, M. and Etzioni, O., “Open Information Extraction from the Web”, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, Hyderabad, India, 2007.

[12] Gelbaum, C., editor, **WordNet: An Electronic Lexical Database**, MIT Press, Cambridge, MA, 1998.

[13] McCallum, A. & Wellner, B., “Conditional Models of Identity Uncertainty with Application to Noun Coreference, *Advances in Neural Information Processing Systems 17*, 2005.

[14] Haghighi, A. & Klein, D., “Unsupervised Coreference Resolution in a Nonparametric Bayesian Model”, *45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic, 2007.

[15] Poon, H. & Domingos, P., “Sound and Efficient Inference with Probabilistic and Deterministic Dependencies”, *Proceedings of the Twenty First National Conference on Artificial Intelligence*, Boston, MA, 2006.

[16] Kautz, H.; Selman, B. and Jiang, Y., “A General Stochastic Approach to Solving Problems with Hard and Soft Constraints”, **The Satisfiability Problem: Theory and Applications**. American Mathematical Society, New York, New York, pp. 573-586.

[17] Kok, S.; Sumner, M.; Richardson, M.; Singla, P.; Poon, H.; Lowd , D.; Wang, J. and Domingos, P., “ The Alchemy System for Statistical Relational AI (Technical Report)”, Department of Computer Science and Engineering, University of Washington, 2006.

[18] Wei, W.; Erenrich, J. and Selman B., “Towards Efficient Sampling: Exploiting Random

Walk Strategies", *Proceedings of the Twenty First National Conference on Artificial Intelligence*, 2004.

[19] Vilian, M.; Burger, J; Aberdeen, J.; Connolly, D. & Hirschman, L., "A Model-Theoretic Coreference Scoring Scheme", *Message Understanding Conference*, 1995.

[20] Amit, B. and Baldwin, B., "Algorithms for Scoring Coreference Chains", *Message Understanding Conference*, 1997.

[21] Ng, V., "Machine Learning for Coreference Resolution: From Local Classification to Global Ranking", *43rd Annual Meeting of the Association for Computational Linguistics*, 2005.

[22] Denis, P. & Baldridge, J., "Joint Determination of Anaphoricity and Coreference Resolution using Integer Programming", *Conference of the North American Chapter of the Association for Computational Linguistics*, 2007.

[23] Kok, S. & Domingos, P., "Learning the Structure of Markov Logic Networks", *Proceedings of the Twenty Second International Conference on Machine Learning*, Bonn, Germany, 2005.

[24] Mihalkov, L. & Mooney, R. J., "Bottom-Up Learning of Markov Logic Network Structure", *Proceedings of the Twenty Fourth International Conference on Machine Learning*, Corvallis, Oregon, 2007.

[25] Richards, B. L. & Mooney, R. J., "Learning Relations by Pathfinding", *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1992.

[26] Huynh, T. & Mooney, R. J., "Discriminative Structure and Parameter Learning for Markov Logic Networks", *Proceedings of the Twenty Fifth International Conference on Machine Learning*, Corvallis, Oregon, 2008..

8.0 LIST OF ACRONYMS

ACE	Automatic Content Extraction
BUSL	Bottom-Up Structure Learner
DIESEL	Data Integration and Exploitation System that Learns
i.i.d.	independent and identically distributed
ILP	inductive logic programming
IRM	Infinite Relational Model
ITC	Information-Theoretic Co-clustering
LHL	Learning via Hypergraph Lifting
MAP	maximum a posteriori
MLN	Markov logic network
MRC	Multiple Relational Clustering
MSL	Markov logic Structure Learner
MUC	Message Understanding Conference
NLP	natural language processing
PSCG	preconditioned scaled conjugate gradient
WPLL	weighted pseudo-log-likelihood